ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO BÀI THỰC HÀNH LAB01

HỌ VÀ TÊN: **NGUYỄN GIA BẢO NGỌC – 21520366**
NGUYỄN QUỐC TRƯỜNG AN – 21521810
NGUYỄN ĐỨC LƯU – 21522314

LỚP: **CE224.O12.1**

GIẢNG VIÊN HƯỚNG DẪN:

CHUNG QUANG KHÁNH

TP. HỒ CHÍ MINH – Tháng 10 năm 2023

MỤC LỤC:

Danh mục ảnh:

LAB 1: LÀM QUEN VỚI KIT STM32F4 DISCOVERY

1. CHUẨN BỊ :

(đã thực hiện trong giờ thực hành)

2. HƯỚNG DẪN:

(đã thực hiện trong giờ thực hành)

3. BÀI TẬP:

Bài tập 1: Trình bày về các đặc điểm Timer trên KIT STM32F4 Discovery (số lượng Timer, các chế độ hoạt động, cách cấu hình, cách cài đặt):

a) Số lượng các Timer:

Thiết bị bao gồm 2 “advanced-control timers”, 8 “general-purpose timers” và 2 “basic timers”. Tất cả các bộ đếm đều có thể bị đóng băng ở chế độ “debug”.

Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Complementary output	Max interface clock (MHz)	Max timer clock (MHz) (1)
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	90	180
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	45	90/180
	TIM9	16-bit	Up	Any integer between 1 and 65536	No	2	No	90	180
	TIM10 , TIM11	16-bit	Up	Any integer between 1 and 65536	No	1	No	90	180
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	45	90/180
	TIM13 , TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	45	90/180
Basic	TIM6, TIM7	16-bit	Up	Any integer between 1 and 65536	Yes	0	No	45	90/180

Hình 1 - Đặc điểm của các bộ định thời

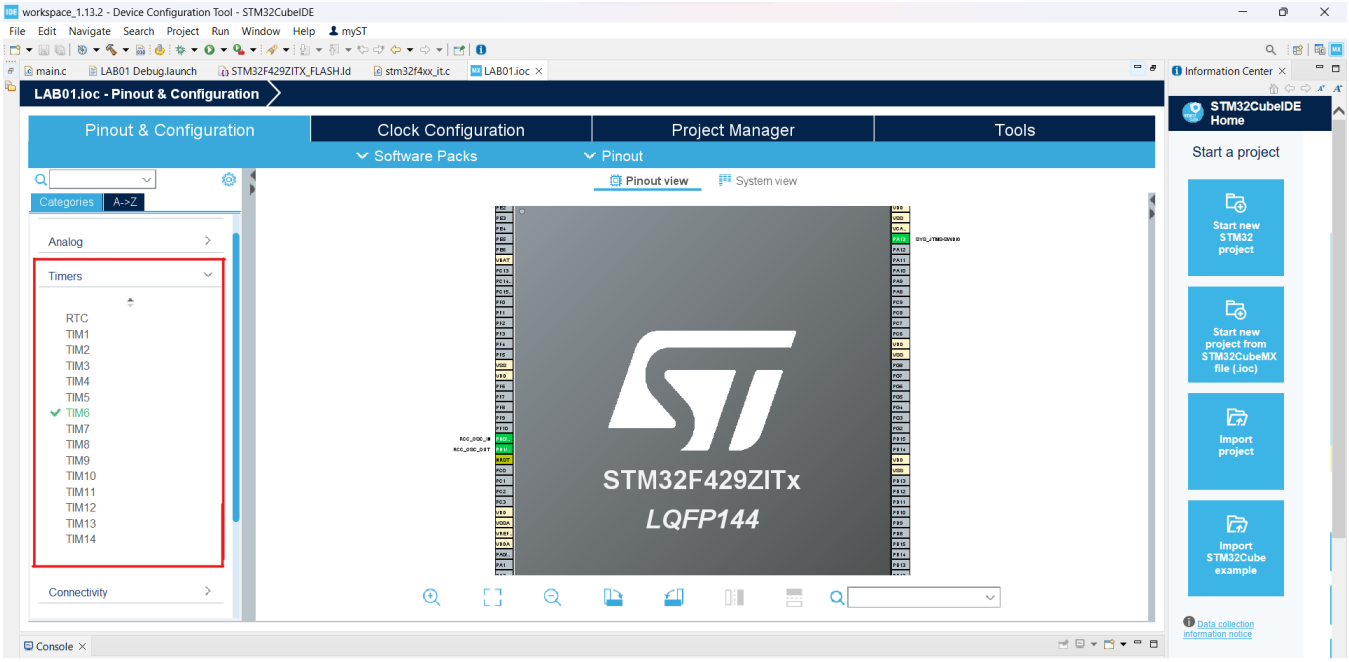
b) Các chế độ định thời:

Bộ định thời của STM32 có thể hoạt động ở nhiều chế độ, nhưng không phải bộ định thời nào cũng có thể hoạt động ở tất cả các chế độ. Như đã trình bày ở trên, chúng ta có các bộ định thời được phân thành các nhóm như: “advanced-control timers”, “general-purpose timers”, “basic timers”. Mỗi nhóm bộ định thời thường hỗ trợ đa chế độ, tuy nhiên vài bộ định thời khác thì có thể hỗ trợ gần như toàn bộ các chế độ hoạt động. Dưới đây chúng ta sẽ đi vào chi tiết hơn các chế độ định thời:

- **Chế độ định thời (Timer):** Trong chế độ định thời, bộ định thời nhận xung “clock” nội với tần số biết trước, nhờ đó thời gian tràn có thể được tính toán và kiểm soát để có thể đạt được thời gian tràn mong muốn từ đầu.
- **Chế độ bộ đếm (Counter):** Trong chế độ bộ đếm, bộ định thời nhận xung “clock” ngoại. Bộ định thời đếm lên hoặc đếm xuống sẽ dựa trên cạnh lên hoặc cạnh xuống của xung “clock” ngoại.
- **Chế độ PWM (Pulse Width Modulation):** Trong chế độ PWM bộ định thời được định thời từ nguồn xung “clock” nội và tạo ra sóng điện tử ở kênh ngõ ra được gọi là tín hiệu PWM.
- **Chế độ PWM nâng cao (Advanced PWM Mode):** là chế độ nâng cao hơn của chế độ PWM, xung tín hiệu PWM nâng cao liên quan đến khả năng phần cứng với mục đích kiểm soát nhiều thông số hơn và thêm vào đó là một số mạch điện tử nhằm hỗ trợ một số tính năng khác cho tín hiệu PWM.
- **Chế độ so sánh ngõ ra (Output Compare Mode):** trong chế độ so sánh ngõ ra, bộ định thời sẽ kiểm soát giá trị đầu ra cho đến khi phát hiện sự trùng khớp của giá trị hiện tại của bộ định thời và giá trị được đặt trước trong thanh ghi giá trị ban đầu (OCR- Output Compare Register). Khi đó một hành động nào đó sẽ được thực thi theo mong muốn của người lập trình (giá trị ngõ ra sẽ được đặt ở mức cao, mức thấp, lật lại so với chính nó ở hiện tại hoặc không có thay đổi gì).
- **Chế độ một xung (One-Pulse Mode):** chế độ là một trường hợp cụ thể của các chế độ trên, ở chế độ một xung cho phép bộ định thời tạo ra một xung có độ dài có thể lập trình được.
- **Chế độ bắt ngõ vào (Input Capture Mode):** trong chế độ bắt ngõ vào, bộ định thời được cấu hình để ghi lại thời điểm của một sự kiện xảy ra. Sự kiện này thường là sự kiện liên quan đến các tín hiệu ngõ vào hay các tín hiệu ngoại vi khác.
- **Chế độ mã hóa (Encoder Mode):** chế độ mã hóa là một chế độ đặc biết của bộ định thời được thiết kế để đọc giá trị từ bộ mã hóa hoặc bộ mã động cơ.
- **Chế độ Timer DMA Burst Mode:** chế độ sử dụng chức năng DMA (Direct Memory Access) của stm32 để truyền dữ liệu. Khi chế độ Timer DMA Burst được kích hoạt, bộ định thời có khả năng truyền dữ liệu theo chế độ Brust vào bộ nhớ thông qua DMA.

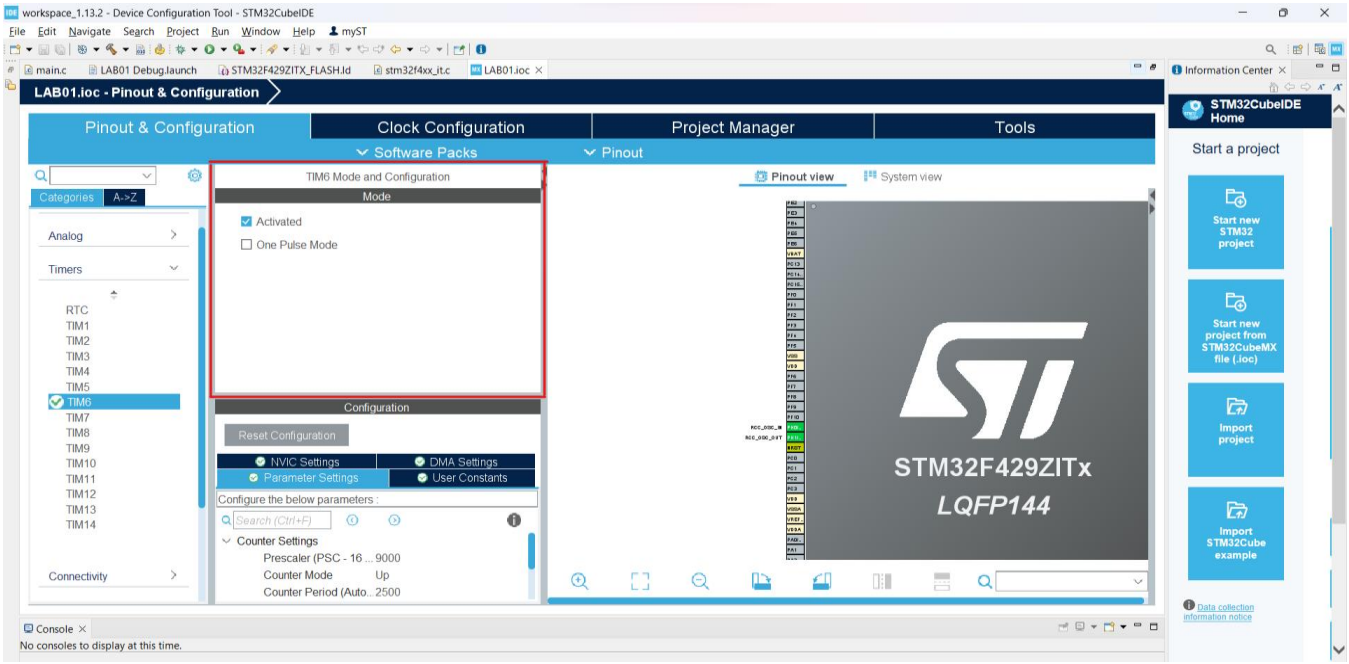
c) Cách cấu hình và cài đặt cho Timer:

- Bước 1: bên góc trái cửa sổ làm việc ta chọn vào mục Timer và chọn vào timer mà mình muốn cấu hình, trong hình timer được chọn là timer số 6.



Hình 2 - Chọn Timer muốn sử dụng

- Bước 2: Chọn chế độ hoạt động cho Timer (chỉ chọn được các chế độ mà Timer được hỗ trợ)

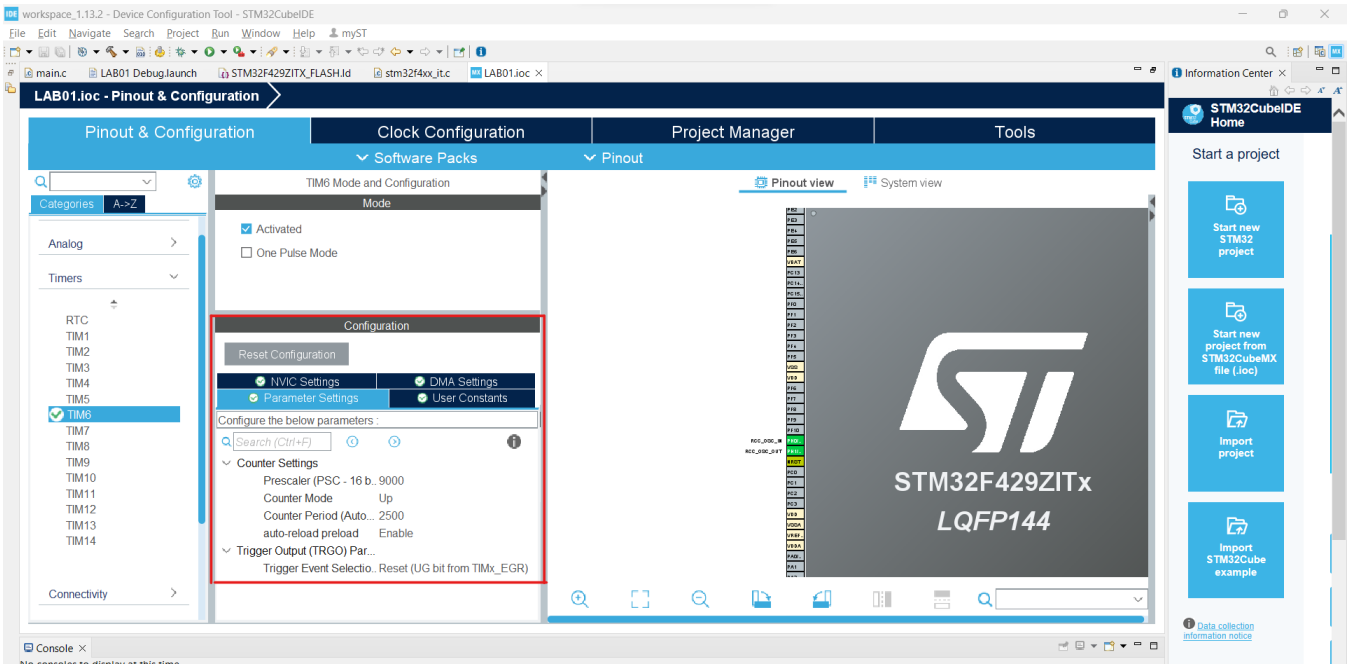


Hình 3 - Chọn chế độ hoạt động cho Timer

- Bước 3: Tại mục Configuration, ta có thể tính toán thời gian tràn cho Timer theo công thức như sau:

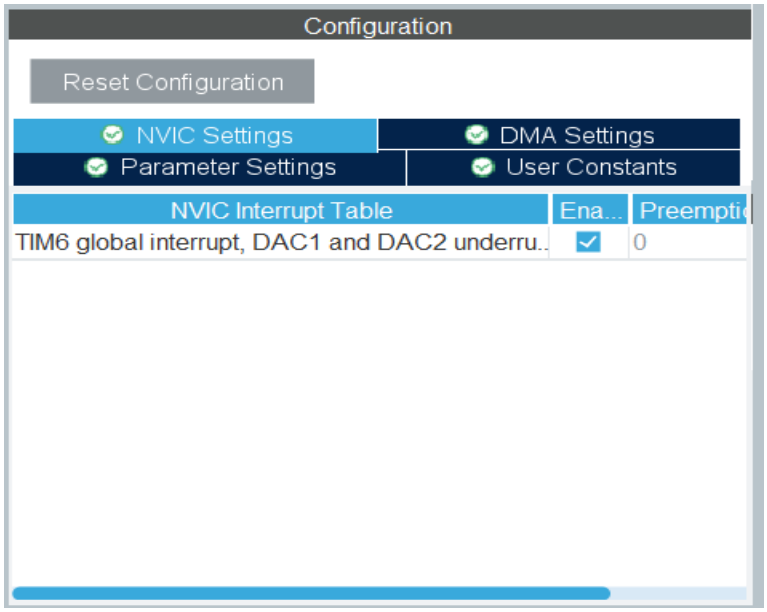
$$F_{\text{Timer}} = \frac{F_{\text{hệ thống}}}{\text{Prescaler} + 1}$$

$$T_{\text{overflow}} = \frac{1}{F_{\text{Timer}}} \times (\text{Counter Period} + 1) = \frac{(\text{Prescaler} + 1) \times (\text{Counter Period} + 1)}{F_{\text{hệ thống}}}$$



Hình 4 - Tính toán thời gian tràn cho Timer

Ngoài ra tại mục NVIC Settings, ta có thể bật ngắt cho Timer



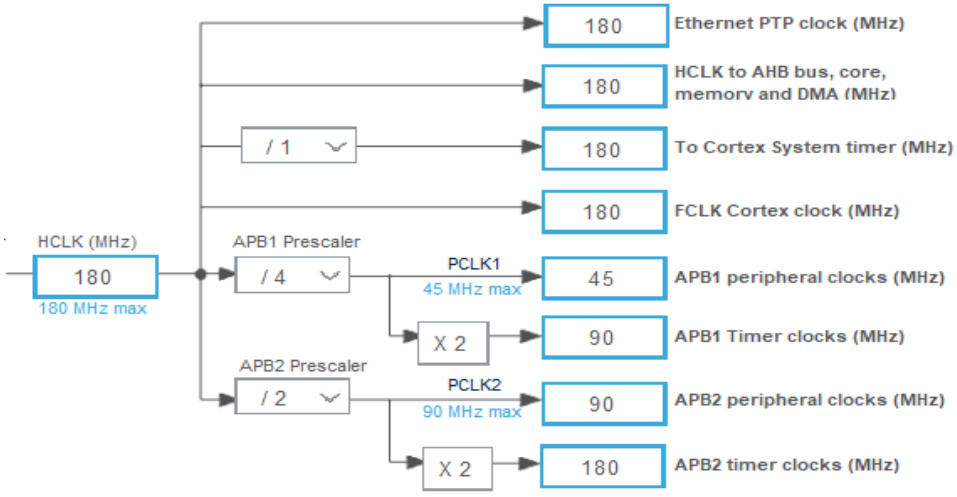
Hình 5 - Bật ngắt cho Timer

Bài tập 2: Sử dụng Timer để viết chương trình chạy 2 LED 3 và LED 4 cùng lúc như sau:

- **LED 3 chớp/tắt với chu kì 01 giây**
- **LED 4 chớp/tắt với chu kì 750ms**

a) Xác định Timer cần dùng và xung nhịp xung clock

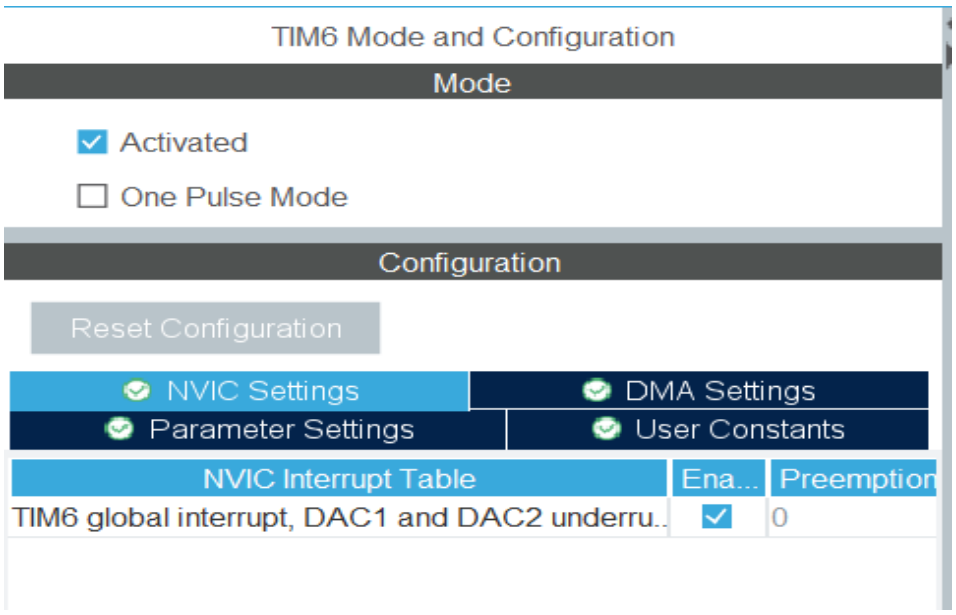
- Ta sử dụng một **Timer 6** để thực hiện yêu cầu. **Timer 6** lấy nhận xung clock từ nguồn **APB1 Timer clocks** có xung nhịp là **90MHz.**



Hình 6 - Xác định xung nhịp clock cho Timer 6

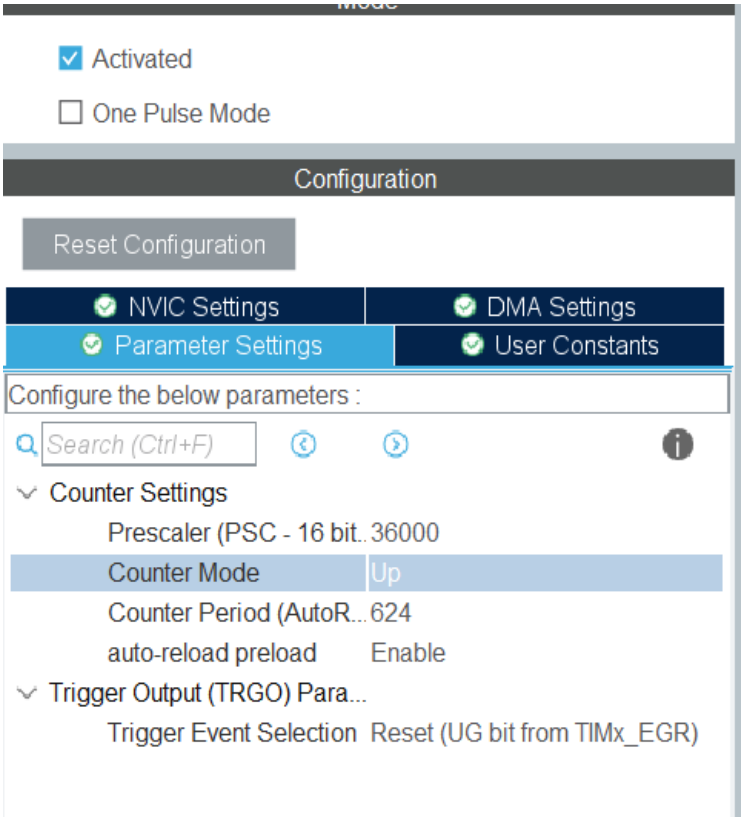
b) Thực hiện cấu hình Timer và Ngắt

- Thực hiện **Activated** Timer 6 và bật **TM6 global interrupt**



Hình 7 - Kích hoạt Timer 6 và Vector Ngắt của Timer 6

- **Cấu hình** tham số Timer như công thức đã trình bày trên bài tập 1 với thông số sau ta có được chu kì tràn của Timer là: **250ms**



Hình 8 - Cấu hình các parameters cho Timer 6

c) Code nạp chạy chương trình

❖ Ở đây nhóm em chỉ trình bày phần code chúng em code thêm vào (**không bao gồm phần code phần mềm đã tổng hợp)**

- Khởi động Timer 6 và vector Ngắt của Timer 6

```
91   /* USER CODE BEGIN 2 */
92   HAL_TIM_Base_Start_IT(&htim6);
93   /* USER CODE END 2 */
```

- Khai báo biến **static cnt** để đếm số lần Timer tràn (dùng biến cnt này trong hàm Ngắt)

```
59 /* USER CODE BEGIN EV */
60 static uint16_t cnt = 0;
61 /* USER CODE END EV */
```

Ta khai báo kiểu **static** để khi thoát ra khỏi hàm ngắt biến cnt vẫn giữ nguyên giá trị

- Hàm xử lý Ngắt khi Timer tràn:
  + Ý tưởng là sau mỗi lần Timer tràn, hàm ngắt được gọi thì sẽ tăng giá trị biến cnt lên 1 (cnt ++). Nếu cnt == 3 tức 250ms * 3 = 750ms thì bật/tắt LED 4. Nếu cnt == 4 tức 250ms * 4 = 1s thì bật/tắt LED 3 và gán cnt về lại 0 (cnt = 0).

```
204 void TIM6_DAC_IRQHandler(void)
205 {
206   /* USER CODE BEGIN TIM6_DAC_IRQn 0 */
207     cnt++;
208     if(cnt == 3){
209         HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_14);
210 
211     } else if(cnt == 4){
212         HAL_GPIO_TogglePin(GPIOG, GPIO_PIN_13);
213         cnt = 0;
214     }
215   /* USER CODE END TIM6_DAC_IRQn 0 */
216   HAL_TIM_IRQHandler(&htim6);
217   /* USER CODE BEGIN TIM6_DAC_IRQn 1 */
218 
219   /* USER CODE END TIM6_DAC_IRQn 1 */
220 }
```